# Tools to Improve Your Software's Performance

## Two Tools, One Name: PerfTools

Brett Viren

Physics Department

BROOKHAVEN
NATIONAL LABORATORY

MINOS Week In The Woods, 2005

# Outline



BROOKHAVEN NATIONAL LABORATORY

BROOKHAVEN NATIONAL LABORATORY

# How they work

Both profilers work the same way:

1. They insinuate code into your program that periodically (many times a second) takes a snapshot of the currently executing call stack.
2. This data is post processed to produce a weighted, directed graph.
3. GraphViz is then used to produce graphical output.

The Graph:

- Each node represents a procedure (function/method) call and shows the frequency that the procedure was:
  - anywhere in the call stack
  - at the top of the call stack
- Each edge represents one procedure calling another and shows the frequency of at which this call was seen.

BROOKHAVEN NATIONAL LABORATORY

# Using the Graphs to Find CPU "Hot Spots"

The graph not only shows what procedure is using CPU but also exactly how that procedure came to be called. Finding possible areas of improvement is an iterative process:

1. Find where the code is using CPU most (traditional profiling)
   - If the procedures using the most CPU can be improved, the whole program is made faster
   - However, these procedures are often already very well optimized (eg. lowlevel STL operations are typically implicated)
   - Instead, they are using the most CPU, not because they are slow, but because they are called often, likely by many different procedures.
2. Go up the graph to find what procedures call these popular procedures most frequently
   - See if it is possibe to avoid calling these procedures in the first place.
   - If not, can one avoid calling the procedures that called **these** procedures?
   - Etc.

BROOKHAVEN NATIONAL LABORATORY

BROOKHAVEN NATIONAL LABORATORY

# Overview of Minos PerfTools (MPT)

- Initially written by Jim Kowalkowski of FNAL. Jim provided all the fundamental functionality.
- I wrote a unified script to automate the post processing and to provide an interactive display to view and prune the resulting graph.
- PerfTools was incorporated into Minos CVS Sept 2003.
- External Requirements:
  1. The ELFIO library is needed for profile data production.
  2. Python is needed for Jim's post processing scripts and my unified post-processor/display.
  3. GraphViz and Python-Tk is needed for the display.
  4. SWIG is needed only to regenerate the GraphViz Python wrappers.

BROOKHAVEN NATIONAL LABORATORY

# Example running

```
shell$ ptrun.py -c ./cpuhog -l log
Running LD_PRELOAD=libProfLogger.so ./cpuhog > log 2>&1
return code: 0
```

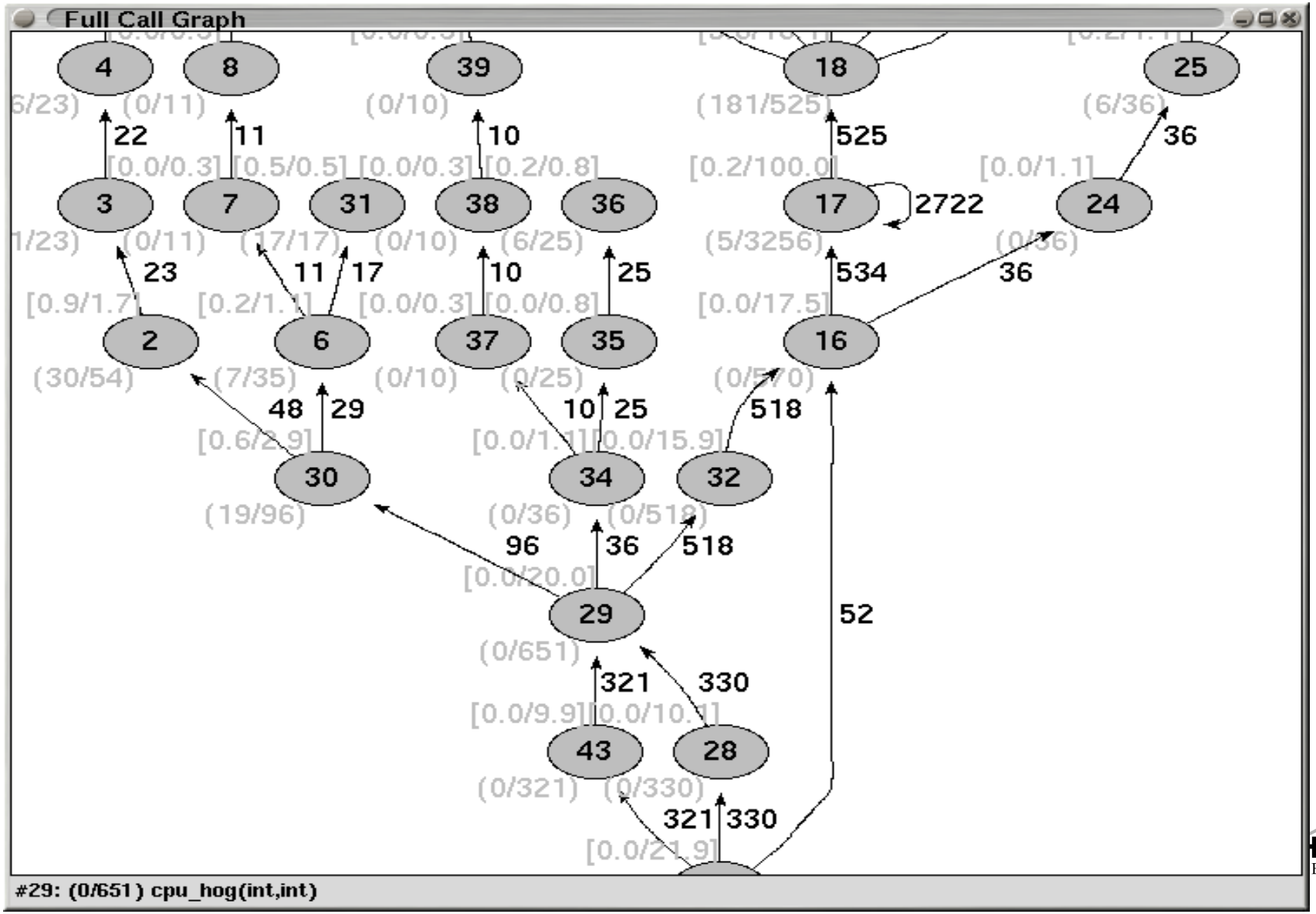
```
shell$ ptrun.py -n guess -b cpuhog
got -n guess
guessing pid
using 24689
Running ProfParse prof_libs.out.24689 prof.out.24689 \
                  cpuhog. >> cpuhog.postproc-log 2>&1
```

```
shell$ ptrun.py -g -b cpuhog
Setting initial path cutoff to 0
Loading cpuhog.fullnames and cpuhog.paths
Added 57 names
Added 265 paths
Number of snapshots taken is 3276
```

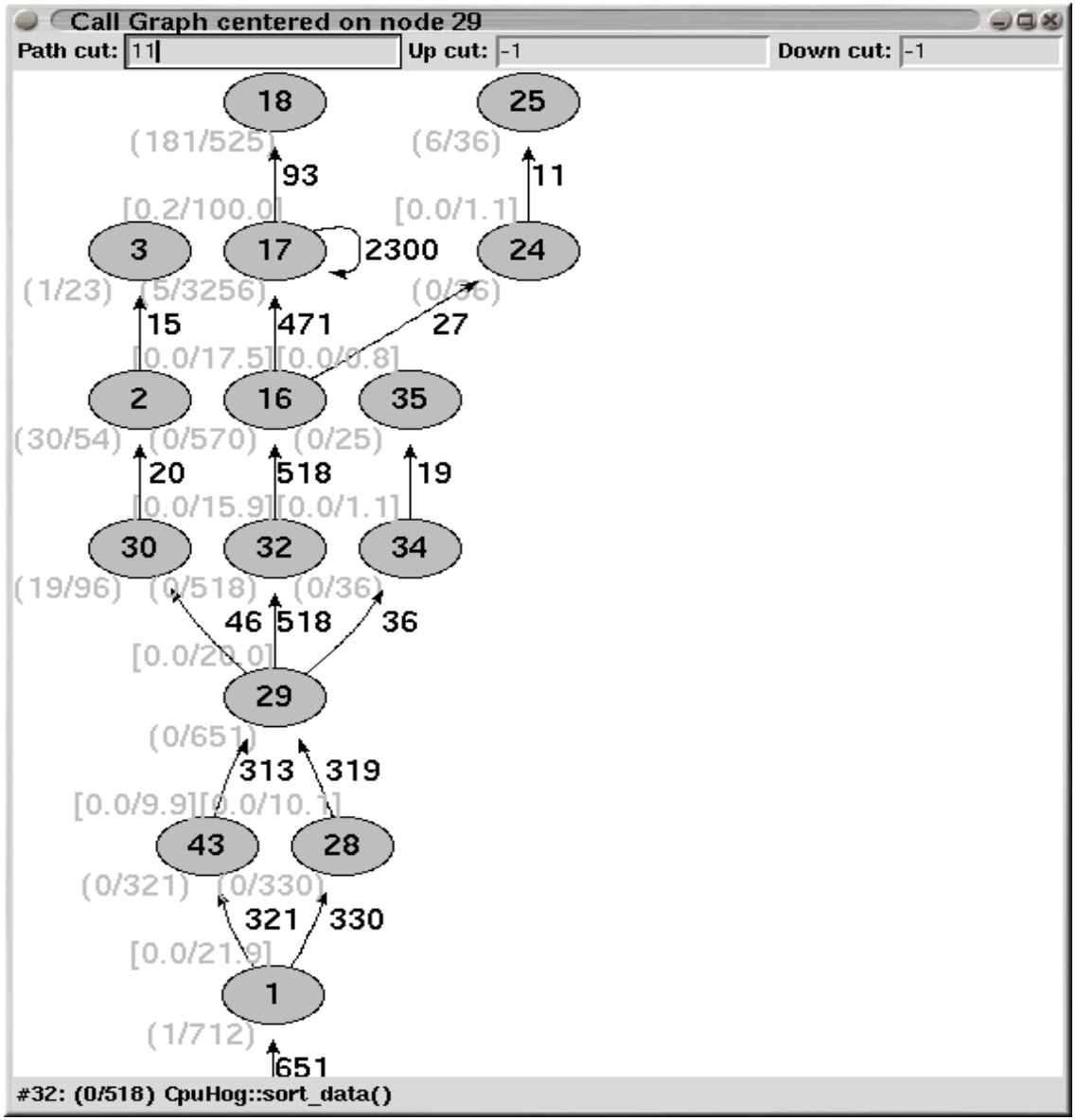
BROOKHAVEN NATIONAL LABORATORY

The 3 steps can be combined into one single command.

# Example output, Full Call Graph



# Example output, After Focus on Node 29




Focusing on Node #29

1. Mouse-over a node displays function name
2. Click-drag pans
3. Buttons 4/5 (mouse wheel) will zoom
4. Enter cuts on number of up/down stream nodes and number of calls.

BROOKHAVEN NATIONAL LABORATORY

BROOKHAVEN NATIONAL LABORATORY

# Overview of Google PerfTools (GPT)

- Initially released to the public March 2005.
- Free Software (BSD licence)
- Despite that the name and the CPU profiling mechansim are essentially identical, GPT share no MPT code.
- Small patch may needed to fix problem when running with Ioon. Under investigation by Google and myself.
- Libraries can be (optionally) permanently linked in to our code w/no ill effects and "woken up" for use by just setting environment variables.
- External Requirements, only "standard" system packages:
  1. Perl for post processing of the data
  2. GraphViz executables to generate final results
  3. Postscript viewer.
  4. (independent from ELFIO)

BROOKHAVEN NATIONAL LABORATORY

# Example Running

The profiling library must be linked into the executable. This can be done in one of two ways:

1. Explicitly by adding `-lprofiler` to the link line
2. At run time by setting `LD_PRELOAD=/usr/lib/libprofiler.so.0`

The profile output file is set via `CPUPROFILE=prof.out`. Then run the program normally.

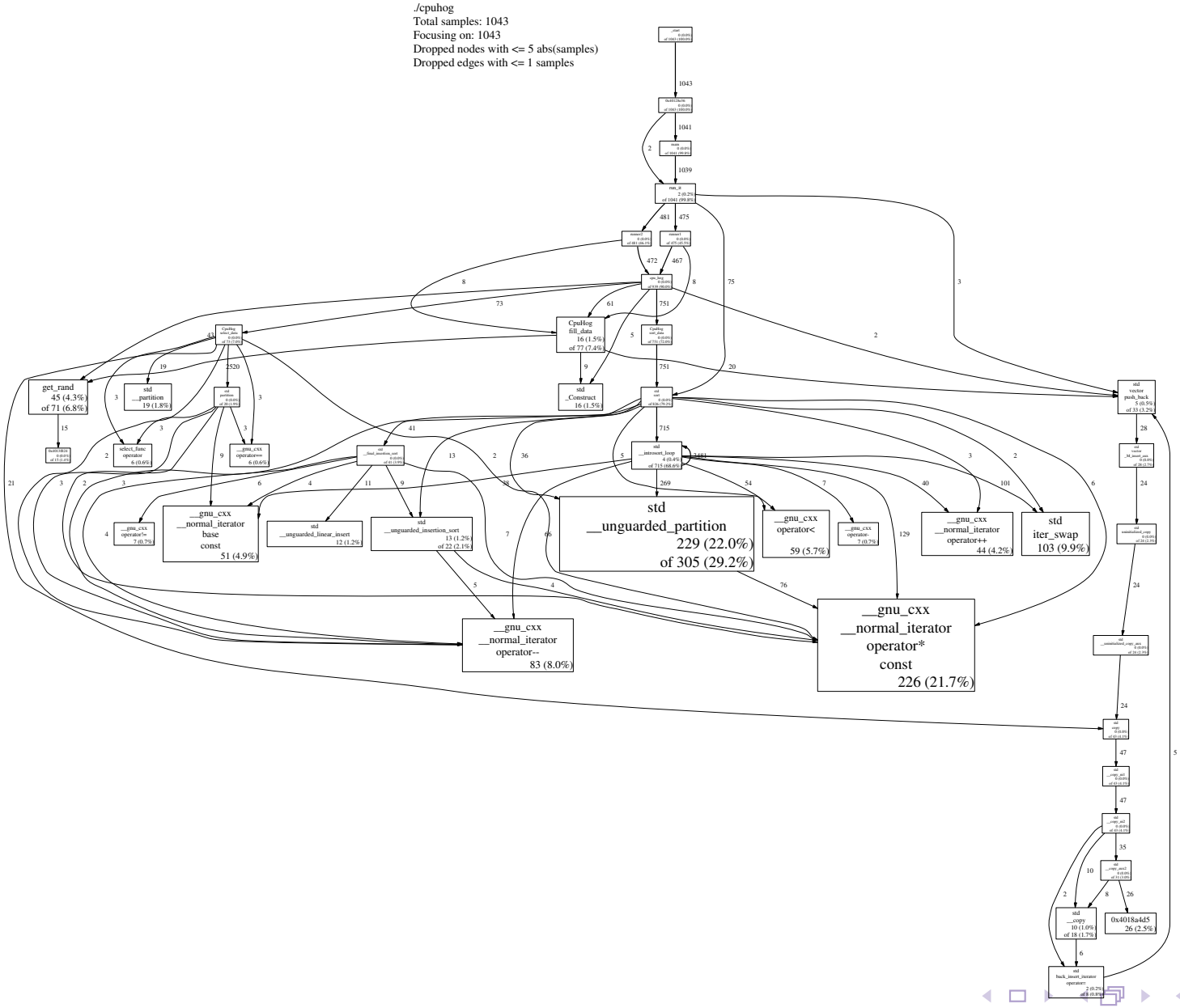
```
shell$ ./cpuhog
...program output...
PROFILE: interrupts/evictions/bytes = 1043/86/52896
```

Postprocess:

```
shell$ pprof --ps ./cpuhog prof.out > prof.ps
...output from /usr/bin/nm calls...
Dropping nodes with <= 5 samples; edges with <= 1 abs(samples)
```

```
shell$ gv prof.ps
```

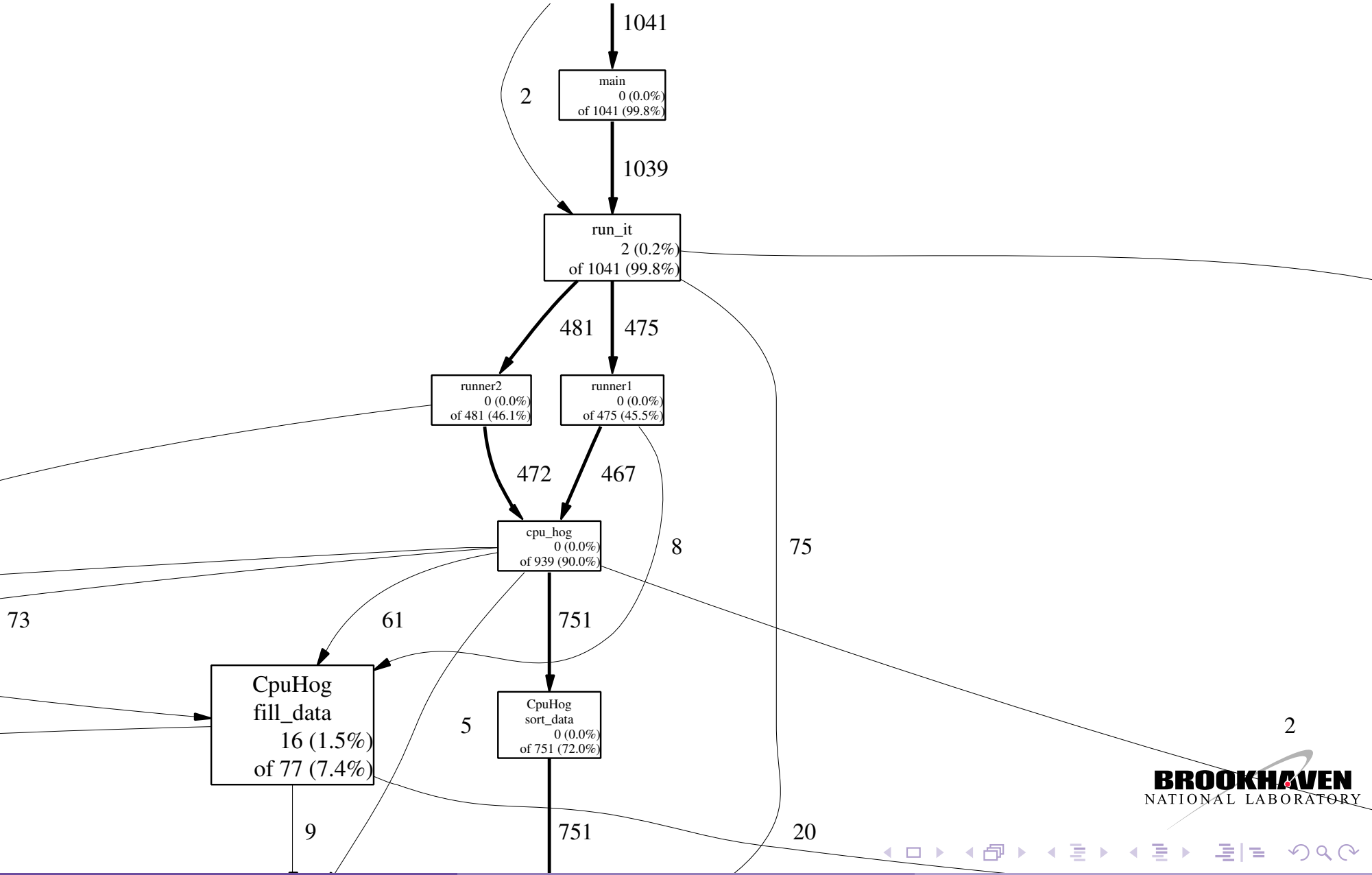
# Example Output



# Example Output

```
./cpuhog
Total samples: 1043
Focusing on: 1043
Dropped nodes with <= 5 abs(samples)
Dropped edges with <= 1 samples
```

# Example Output



# Heap Memory

Leak Checker:

1. Use the `tcmalloc` library as you would the `profiler` lib.
2. Set `HEAPCHECK=normal`, `strict` or `draconian`.

Memory Profiler (who uses how much memory)

1. Use the `tcmalloc` library as you would the `profiler` lib.
2. Set `HEAPPROFILE=heap`

Both can be alternatively turned on/off via function calls in your code.

BROOKHAVEN NATIONAL LABORATORY

# Summary

- We have two similar profilers available.
- Each have benefits and deficits w.r.t. the other.
- When this method of profiling has been applied to MinosSoft code factors of 2-3x speed ups have been found.
- We (core and physics programmers alike) should profile our code regularly after new, significant development has been done.

Personally I much prefer Google's PT. After some of the bugs in GPT are worked out, I will consider our PT no longer supported.

BROOKHAVEN NATIONAL LABORATORY

# For Further Information I

- Google Perftools page
  `http://goog-perftools.sourceforge.net/`
- Minos PerfTools page
  `http://minos.phy.bnl.gov/software/prof/PerfTools/doc/`
- GraphViz (recently became Free Software!)
  `http://www.graphviz.org/`
- Simplified Wrapper and Interface Generator (SWIG)
  `http://www.swig.org/`

BROOKHAVEN NATIONAL LABORATORY